Universidad Autónoma del Estado de Hidalgo

REPORTE DE PRÁCTICA NO. 1

NOMBRE DE LA PRÁCTICA:LENGUAJES FORMALES Y AUTÓMATAS

ALUMNO:

Linares Carrada Jasiel

1. Introducción

Los lenguajes formales constituyen la intersección fundamental entre las matemáticas y la lingüística, proporcionando herramientas para estudiar los números como si se tratasen de palabras. Como se explica en el [1], esta disciplina es la base de la teoría de autómatas y permite analizar propiedades como la de los números capicúa (que se leen igual de izquierda a derecha que de derecha a izquierda). Los lenguajes formales nos permiten navegar entre infinitos y posibilidades, definiendo estructuras matemáticas precisas para el procesamiento de información. Este reporte integra los conceptos fundamentales de los 10 videos analizados, desde los fundamentos de lenguajes formales hasta aplicaciones avanzadas en pattern matching.

2. Marco teórico

Alfabetos y palabras

Un alfabeto es un conjunto finito de símbolos que constituye la base de un lenguaje formal [1]. Por ejemplo, el alfabeto español contiene las letras de la A a la Z, mientras que el alfabeto binario contiene los símbolos 0 y 1. Una palabra (o cadena) es una secuencia finita y ordenada de símbolos de un alfabeto. Por ejemplo, "codem" es una palabra válida sobre el alfabeto español, pero no sobre el alfabeto binario. Un concepto fundamental es la cadena vacía (o), que es una cadena de longitud cero que no contiene ningún símbolo y actúa como elemento neutro en diversas operaciones.

Operaciones con palabras

Las operaciones básicas con palabras incluyen [2]:

- **Concatenación**: Unión de dos palabras. Ejemplo: "monta" + "puercos" = "montapuercos"
- **Potencia**: Repetición de una palabra n veces. Ejemplo: "code"^{3} = "codecodecode"
- **Prefijos, sufijos y segmentos**: Partes de una palabra. Para "100", los prefijos son , 1, 10, 100, los sufijos son , 1, 01, 001, 100
- **Reverso**: Palabra leída al revés. Ejemplo: reverso("Hola") = "aloH"

Lenguajes formales

Un lenguaje formal es un subconjunto de la clausura de Kleene (*) sobre un alfabeto [2]. Puede ser finito o infinito. Ejemplos:

- Lenguaje de palabras que empiezan por "a": L = ax donde x *
- Lenguaje de palabras de longitud menor que 5: L = x * — —x— ¡ 5

Operaciones con lenguajes

Las operaciones booleanas (unión, intersección, complemento) se aplican a lenguajes como a cualquier conjunto [3]. Operaciones específicas incluyen:

- **Producto**: Concatenación de cada palabra de un lenguaje con cada palabra de otro
- **Potencia y cierre**: Extensiones de las operaciones con palabras a nivel de lenguajes
- **Cociente**: Eliminación de prefijos o sufijos de las palabras de un lenguaje
- **Homomorfismo**: Transformación de símbolos entre alfabetos diferentes

Autómatas finitos

Un autómata es una máquina abstracta que procesa inputs y devuelve aceptación o no aceptación [4]. Los autómatas finitos deterministas (AFD) se definen por la tupla (Q, , , q, F) donde:

- Q: Conjunto finito de estados
- : Alfabeto de entrada
- : Función de transición Q × → Q
- q: Estado inicial
- F: Conjunto de estados finales

Autómatas no deterministas

Los autómatas finitos no deterministas (AFND) permiten múltiples transiciones desde un estado con el mismo símbolo [6]. Se definen por (Q, , , q, F) donde : Q × → 2 elevado a Q. Todo AFD es un AFND, pero no viceversa.

Autómatas con transiciones vacías

Los AF- permiten transiciones etiquetadas con la cadena vacía (), permitiendo cambiar de estado sin procesar símbolos [8]. Estos autómatas pueden convertirse en AFND equivalentes mediante el cálculo de -clausuras.

Pattern matching

El pattern matching o coincidencia de patrones es fundamental en informática para encontrar ocurrencias de patrones en textos [10]. Dos algoritmos importantes son:

- **Algoritmo ingenuo**: Utiliza un árbol aceptor de prefijos con coste potencialmente cuadrático
- **Algoritmo del autómata diccionario**: Utiliza sufijos para un preprocesamiento más eficiente con coste lineal

3. Herramientas empleadas

1. **Teoría de conjuntos**: Base matemática para definir alfabetos, palabras y lenguajes
2. **Grafos**: Para representar visualmente autómatas y sus transiciones
3. **Notación matemática formal**: Para definir precisamente los componentes de autómatas
4. **Algoritmos de conversión**: Para transformar entre diferentes tipos de autómatas

4. Desarrollo

Ejemplo 1: Autómata finito determinista para números binarios

Se diseñó un AFD que acepta números binarios que contienen al menos un 0. El autómata se define formalmente como:

- **Estados**: Q = q, q
- **Alfabeto**: = 0, 1
- **Estado inicial**: q
- **Estados finales**: F = q
- **Función de transición**:
  - (q, 0) = q
  - (q, 1) = q
  - (q, 0) = q
  - (q, 1) = q

Este autómata acepta palabras como "0", "10", "010", "1010" pero rechaza "1", "11", "111". La representación gráfica muestra claramente cómo desde el estado inicial q (no final) se permanece en q con el símbolo 1, pero al recibir un 0 se transita al estado final q, donde se permanece indefinidamente con cualquier secuencia posterior de 0s y 1s.

Ejemplo 2: Conversión de AFND a AFD

Se implementó el algoritmo de conversión para transformar el siguiente AFND a su AFD equivalente:
AFND original:

- **Estados**: Q = q, q, q
- **Alfabeto**: = a, b
- **Estado inicial**: q
- **Estados finales**: F = q
- **Función de transición**:
  - (q, a) = q, q
  - (q, b) = q
  - (q, a) = q
  - (q, b) = q
  - (q, a) = q
  - (q, b) = q

Mediante el método de construcción de subconjuntos, se obtuvo el AFD equivalente:

- **Estados**: Q' = q, q, q, q, q, q
- **Estado inicial**: q
- **Estados finales**: F' = q, q, q (todos los estados que contienen q)
- **Función de transición**:

- ’(q, a) = q, q
- ’(q, b) = q
- ’(q, q, a) = q, q, q
- ’(q, q, b) = q, q
- ’(q, q, q, a) = q, q, q
- ’(q, q, q, b) = q, q
- ’(q, q, a) = q, q, q
- ’(q, q, b) = q, q

Esta conversión demuestra cómo un autómata no determinista con 3 estados puede convertirse en un autómata determinista equivalente con 4 estados, preservando exactamente el mismo lenguaje reconocido.

5. Conclusiones

El estudio de lenguajes formales y autómatas proporciona herramientas fundamentales para el procesamiento eficiente de información. Los conceptos de alfabetos, palabras y lenguajes [1, 2] sentaron las bases para comprender estructuras más complejas como los autómatas finitos [4, 5]. La distinción entre autómatas deterministas y no deterministas [6] revela diferentes enfoques para resolver problemas de reconocimiento de patrones.

Un momento "WOW" fue comprender cómo los autómatas con transiciones vacías [8] pueden convertirse en autómatas equivalentes sin estas transiciones, manteniendo su poder computacional. Las aplicaciones en pattern matching [10] demostraron la utilidad práctica de estos conceptos teóricos, mostrando cómo el preprocesamiento adecuado puede reducir la complejidad algorítmica de cuadrática a lineal.

La teoría de autómatas y lenguajes formales continúa siendo relevante en áreas como compiladores, procesamiento de lenguaje natural y verificación formal de software, demostrando que los fundamentos teóricos establecidos hace décadas siguen enabling avances tecnológicos contemporáneos.

Referencias Bibliográficas

References

[1] Canal de Lenguajes Formales. (2023). *Introducción a los lenguajes formales: Alfabetos y palabras* [Video]. YouTube. https://youtu.be/_UdVL-84rXc?si=Zs7SrEkDkI9Rz3MA

[2] Canal de Lenguajes Formales. (2023). *Operaciones con palabras y definición de lenguajes* [Video]. YouTube. https://youtu.be/MXDl4Ts_EZ0?si=vm_FcIcgdVgmOojY

[3] Canal de Lenguajes Formales. (2023). *Operaciones con lenguajes formales* [Video]. YouTube. https://youtu.be/uU-fNuwbmZg?si=pagfNDOZqySlOjjv

[4] Canal de Autómatas. (2023). *Introducción a los autómatas* [Video]. YouTube. https://youtu.be/pMIwci0kMv0?si=mmHN-gSXu24OdnBp

[5] Canal de Autómatas. (2023). *Autómatas finitos deterministas (AFD)* [Video]. YouTube. https://youtu.be/d9aEE-uLmNE?si=JsItPkLMacFsJSd1

[6] Canal de Autómatas. (2023). *Autómatas finitos no deterministas (AFND)* [Video]. YouTube. https://youtu.be/dIgKBNuaglE?si=ABSDA6qFIvP1GYyl

[7] Canal de Autómatas. (2023). *Conversión de AFND a AFD* [Video]. YouTube. https://youtu.be/hzJ8CNdPElc?si=TutNvEEsa_REey7B

[8] Canal de Autómatas. (2023). *Autómatas con transiciones vacías (AF-)* [Video]. YouTube. https://youtu.be/71P3daDZWlQ?si=XqakDHcsRlQ0-xO6

[9] Canal de Autómatas. (2023). *Conversión de AF- a AFND* [Video]. YouTube. https://youtu.be/1yKBT8gWN-Y?si=VsH9aHmwM7P2IyGK

[10] Canal de Pattern Matching. (2023). *Pattern matching con autómatas* [Video]. YouTube. https://youtu.be/1yKBT8gWN-Y?si=66SmNEaVkG1CdpxC

[11] Canal de Ensamblador. (2023). *Introducción al lenguaje ensamblador* [Video]. YouTube. https://youtu.be/JuTuMe8Q58c?si=bkXpOYdYjCHAsDmJ

[12] Canal de Ensamblador. (2023). *Instrucciones aritméticas y lógicas* [Video]. YouTube. https://youtu.be/gYOvlrjRBwg?si=-Ri5f5V222Bza2Wz

[13] Canal de Ensamblador. (2023). *Interrupciones y manejo de E/S* [Video]. YouTube. https://youtu.be/FPWpCq20g0o?si=xfI7OkCD3MOuP2L6

[14] Canal de Optimización. (2023). *Optimización y buenas prácticas* [Video]. YouTube. https://youtu.be/gd6uyNXsqcw?si=VWw_Nkl4Dh240Dzs

[15] Canal de Compiladores. (2023). *Introducción a compiladores* [Video]. YouTube. https://youtu.be/x4CugLUufTc?si=uZfBrxQhWqX-i0A9

[16] Canal de Compiladores. (2023). *Análisis léxico y sintáctico* [Video]. YouTube. https://youtu.be/5dmyN5mWu1o?si=do9Sooj2TLnFQIMQ

[17] Canal de Compiladores. (2023). *Generación de código intermedio* [Video]. YouTube. https://youtu.be/qzVms1j23uE?si=HUblHGk0icLtUK9a

[18] Charte Ojeda, F., & Ruíz Calderón, V. M. (2020). *Lenguaje ensamblador*. RA-MA Editorial.

[19] Alfonseca Moreno, M. (1998). *Compiladores e intérpretes: teoría y práctica*. McGraw-Hill.

[20] Aho, A. V., Sethi, R., & Ullman, J. D. (2007). *Compiladores: principios, técnicas y herramientas* (2.a ed.). Pearson Educación.